Sine Qua Non for Large Theory Reasoning

Kryštof Hoder and Andrei Voronkov[⋆]

University of Manchester, Manchester, UK

Abstract. One possible way to deal with large theories is to have a good selection method for relevant axioms. This is confirmed by the fact that the largest available first-order knowledge base (the Open CYC) contains over 3 million axioms, while answering queries to it usually requires not more than a few dozen axioms. A method for axiom selection has been proposed by the first author in the Sumo INference Engine (SInE) system. SInE has won the large theory division of CASC in 2008. The method turned out to be so successful that the next two years it was used by the winner as well as by several other competing systems. This paper contains the presentation of the method and describes experiments with it in the theorem prover Vampire.

1 Introduction

First-order theorem provers traditionally were designed for working with relatively small collections of input formulas, for example, those based on a small axiomatisation of a class of algebras, or some axiomatisation of a set theory. Recently, several very large first-order axiomatisations and problems using these axiomatisations have become available. Problems of this kind usually come either from knowledge-base reasoning over large ontologies (such as SUMO [6] and CYC [4]) or from reasoning over large mathematical libraries (such as MIZAR [9]). Solving these problems usually involves reasoning in theories that contain thousands to millions of axioms, of which only a few are going to be used in proofs we are looking for.

Reasoning with very large theories expressed in first-order logic requires radical redesign of theorem provers. For example, a quadratic time preprocessing algorithm (such algorithms were routinely used in the past) may become prohibitively expensive when the input contains a million formulas.

The first-order problems we will discuss in this paper consist of a very large (thousands to millions) set Ax of *axioms*, plus a small number of additional *assumptions* $A_1, \ldots, A_n$ and a *conjecture* G, sometimes also called a *goal*. We have to prove the conjecture from the axioms and assumptions. Since the set of additional assumptions is normally small (and often empty), it will be convenient for us to assume that we have a large set of axioms and a single goal $A_1 \land \ldots \land A_n \to G$. When we discuss complexity of algorithms in this paper, we assume that all axioms and the goal are *small*, for example, have a size (number of symbols, connectives and quantifiers) bound by a constant.

If the conjecture is provable from the axioms, then it is normally provable from a very small subset of these axioms. For example, some of the CYC problems mentioned

[⋆] This work and the second author are partially supported by an EPSRC grant.

above contain over 3,000,000 axioms, and all of these problems have proofs involving only less than 20 axioms. If we only use the axioms occurring in such a proof instead of all axioms, a proof will be found by any modern theorem prover in essentially no time.

Provided that only a tiny subset T of axioms is sufficient for finding a proof, one can try to select a small subset $S \subseteq \mathsf{Ax}$ of axioms, which is likely to contain T, and search for a proof using a standard first-order theorem prover on the subset S instead of Ax. It is common that the subset S we are trying to select consists of the axioms *most relevant* to the goal. This paper describes an algorithm for axiom selection. The first version of the algorithm was originally introduced by the first author and implemented in the system SInE. The version and options described here are implemented in the theorem prover Vampire [3].

This paper is structured as follows. In Section 2 we discuss the problem of selecting axioms relevant to a goal and the natural idea of *symbol-based selection*. Based on this discussion, in Subsection 2.2 we introduce a definition of *trigger-based selection*, which captures a special case of symbol-based selection. In Section 3 we present the Sine selection algorithm as a trigger-based selection algorithm. In Section 4 we discuss possible variations of this algorithm obtained by changing the trigger relation to overcome potential shortcomings of Sine selection. We also describe the Vampire parameters that can be used to invoke these variations.

Section 5 presents experimental results carried out over TPTP problems with large axiomatisations. It shows the effect of various parameter values on the size of the selected set of axioms, the number of iterations of the algorithm, and on the ability to solve hard TPTP problems. Section 6 describes the use of our selection method in the recent CASC competitions. In Section 7 we briefly overview other algorithms used for selection of relevant axioms and other related work.

2 Symbol-Based Selection

2.1 Idea: Relevance

When one thinks of selecting axioms relevant to a goal, perhaps the most natural idea is to use *symbol-based selection*. By a symbol we mean any predicate or function symbol (including constants) apart from equality $=$. Symbol-based selection means that axioms are selected based on symbols occurring in them. Let us call two symbols *neighbours* if either they occur in the same axiom. Let us also call two symbols s_1, s_2 *relevant* if (s_1, s_2) belongs to the reflexive and transitive closure of the neighbour relation. We will say that a symbol is *relevant* (to the goal) if it is relevant to a symbol occurring in the goal. Likewise, we say that an *axiom is relevant* if it contains at least one relevant symbol. Note that in an axiom containing at least one relevant symbol, all symbols will be relevant too.

One possible way of axiom selection is to use all relevant axioms. However, for all benchmark suites available to us, the set of relevant symbols is usually the set of all or nearly all axioms. This is mostly due to the use of very general relations having occurrences in many axioms: as soon as any such relation is relevant, all axioms containing this relation become relevant. We will refer to symbols having occurrences in many axioms as to *common symbols*. Typical examples of common symbols are "instance-of" and "subclass" relations in ontologies: many ontologies consists almost exclusively of axioms using these two relations (or similar relations, such as "subsumes"). Therefore, any selection procedure that tries to avoid selecting nearly all axioms should solve the problem of common symbols.

Another idea for selecting fewer axioms is not to use the full reflexive and transitive closure of the neighbour relation but only a subset thereof, for example by only allowing to make n steps of the computation of the transitive closure, for some (small) positive integer n. Let us formalise the relevancy relation, so that we can refine it later. More precisely, we will deal with two relevancy relations, one for symbols and another for axioms.

1. If s is a symbol occurring in the goal, then s is 0-step relevant.
2. If s is k-step relevant and occurs in an axiom A, then A is $k + 1$-step relevant.
3. A is k-step relevant and s occurs in A, then s is k-step relevant, too.

Clearly, a symbol or an axiom is relevant, if it is k-relevant for some $k \geq 0$. This definition implies also that a k-relevant symbol or axiom is m-relevant for every $m \geq k$.

One can use this inductive definition to select either the set of all k-relevant, for some fixed k, axioms, or the set of all relevant axioms. To compute the latter, we can mark all relevant axioms until the inductive step does not select any new axiom. Moreover, it is easy to implement this algorithm in the time linear in the size of the set of all axioms.

Even better, assuming that the set of all axioms is fixed (which is a natural assumption for applications), by preprocessing the set of axioms one can compute the set of relevant (or k-step relevant) axioms in the time *linear in the size of the computed set*. To this end, it is sufficient to index the set of all pairs (s, A) such that s is a symbol occurring in an axiom A (of course, A can be represented as a reference in the index). This index can be implemented by storing for every symbol the set of axioms in which it occurs. Let us show that such an algorithm is indeed linear in the size of the computed set. To this end, let us consider the set of pairs (s, A) inspected by this algorithm. Note that for every such pair, A will be included in the output. We use an assumption that the size of every axiom is bound by a constant; therefore the number of symbols occurring in a single axiom is also bound by a constant c. This implies that, for a given axiom A, pairs of the form (s, A) can be inspected at most c times. This shows that if axioms of the total size n were selected, the runtime of the algorithm is in the worst case $O(c \cdot n)$, where c is the maximal number of symbols occurring in an axiom.

Note that using k-step relevance instead of relevance does not solve the problem of common relations, since they can already become relevant for very small values of k.

2.2 General Scheme

We will introduce a class of symbol-based selection algorithms that generalise and refine the idea of symbol-based selection. The only deviation from the previous definition is to add extra conditions for an axiom to be relevant. We will achieve this by introducing a trigger relation as follows. Suppose that we have a relation $\mathit{trigger}(s, A)$ between symbols and axioms, such that this relation holds only if s occurs in A. If this relation holds, we will also say that *s triggers A*.

Definition 1 (trigger-based selection)

1. *If s is a symbol occurring in the goal, then s is 0-step triggered.*
2. *If s is k-step triggered and s triggers A, then A is $k+1$-step triggered.*
3. *A is k-step triggered and s occurs in A, then s is k-step triggered, too.*

An axiom or a symbol is called triggered *if it is k-triggered for some $k \geq 0$.*

It is easy to see that the introduction of the trigger relation can solve the problem of common symbols. One can simply postulate that s triggers A only if s is not a common symbol. Or, to be on a safe side, one can say that a common symbol s triggers A only if all symbols of A are common. This would avoid excluding axioms like

$$\mathit{subclass}(x, y) \wedge \mathit{subclass}(y, z) \rightarrow \mathit{subclass}(x, z)$$

(transitivity of the subclass relation) or

$$\mathit{instanceof}(x, y) \wedge \mathit{subclass}(y, z) \rightarrow \mathit{instanceof}(x, z).$$

It is not hard to argue that if the set of all symbols s such that s triggers axiom A can be computed in the time linear in the size of A, then the set T of all triggered (as well as all k-step triggered) axioms can be computed in time linear in the size of T. This is achieved by keeping a mapping from symbols to the sets of axioms which they trigger.

3 The Sine Selection

3.1 Idea

The Sine selection algorithm is a special case of trigger-based selection. It uses a trigger relation that tries to reflect, in a certain way, the hard-to-formalise notions *"s_2 is defined using s_1"* or *"s_1 is more general than s_2"* on symbols.

It is not unreasonable to assume that large knowledge bases contain large hierarchical collections of definitions, where more general terms are defined using less general terms. It is not easy to extract such definitions, since they can take various forms. It is also not easy to formalise the relation "more general". As a simple approximation to "more general" one can consider the relation "more common": a symbol s_2 is considered more common than s_1 if s_1 occurs in more axioms than s_2. Then, as a potential approximation to *"s_2 is defined in terms of s_1"* we can consider the relation *"s_1, s_2 occur in the same axiom A and s_2 is a least common symbol in A."* This is, essentially, the definition of the trigger function for the Sine selection.

Definition 2 (Trigger relation for the Sine selection). *Let us denote by $occ(s)$ the number of axioms in which the symbol s appears. Then we define the relation $trigger$ as follows: $trigger(s, A)$ iff for all symbols s' occurring in A we have $occ(s) \leq occ(s')$. In other words, an axiom is only triggered by the least common symbols occurring in it.*

3.2 Examples

Example 3 (Sine selection). In this example we will denote variables by capital letters. The example illustrates the Sine selection and also a typical reason why it is, in general, incomplete. Consider the following set of axioms:

```
subclass(X,Y) ∧ subclass(Y,Z) → subclass(X,Z)
subclass(petrol,liquid)
¬subclass(stone,liquid)
subclass(beverage,liquid)
subclass(beer,beverage)
subclass(guinness,beer)
subclass(pilsner,beer)
```

The following table gives, for every symbol s, the number of axioms in which it occurs.

s	$occ(s)$
`subclass`	7
`liquid`	3
`beer`	3
`beverage`	2
`petrol`	1
`stone`	1
`guinness`	1
`pilsner`	1

Using the occurrence table, we can compute the trigger relation as follows:

axiom	symbols
`subclass(X,Y) ∧ subclass(Y,Z) → subclass(X,Z)`	`subclass`
`subclass(petrol,liquid)`	`petrol`
`¬subclass(stone,liquid)`	`stone`
`subclass(beverage,liquid)`	`beverage`
`subclass(beer,beverage)`	`beverage`
`subclass(guinness,beer)`	`guinness`
`subclass(pilsner,beer)`	`pilsner`

Consider the goal `subclass(beer,liquid)`. This goal is a logical consequence of these axioms. However, the symbols from the goal `subclass`, `beer`, and `liquid` only trigger the first axiom. The selection will terminate with only the first axiom selected, which is insufficient to prove the goal. □

Consider another example. This example illustrates how (small) changes in the input set of axioms can influence selection.

Example 4. Let us remove the last axiom from the axioms of Example 3. This changes the function *occ* as follows:

s	$occ(s)$
`subclass`	6
`liquid`	3
`beer`	2
`beverage`	2
`petrol`	1
`stone`	1
`guinness`	1

This also changes the trigger relation as follows:

axiom	symbols
`subclass(X,Y)` $\wedge$ `subclass(Y,Z)` $\rightarrow$ `subclass(X,Z)`	`subclass`
`subclass(petrol,liquid)`	`petrol`
$\neg$`subclass(stone,liquid)`	`stone`
`subclass(beverage,liquid)`	`beverage`
`subclass(beer,beverage)`	`beverage,beer`
`subclass(guinness,beer)`	`guinness`

Consider the same goal `subclass(beer,liquid)` as in Example 3. Now the symbols from the goal `subclass`, `beer`, and `liquid` trigger the first axiom as before plus the axiom `subclass(beer,beverage)`. This results in adding `beverage` to the list of triggered symbols. As a consequence, this addition triggers the axiom `subclass(beverage,liquid)`. This triggers no new symbols (since `beverage` was already triggered before), and so selection terminates with the following subset of selected axioms

```
subclass(X,Y) ∧ subclass(Y,Z) → subclass(X,Z)
subclass(beverage,liquid)
subclass(beer,beverage)
```

This collection of axioms is sufficient to prove the goal, contrary to Example 3. Moreover, it is the minimal set of axioms sufficient to prove this goal. □

This example also illustrates that removing some axioms from the input set can result in selecting more axioms than before the removal.

3.3 The Selection Algorithm in More Detail

The algorithm runs in two phases. The first phase is goal-independent and only preprocesses the set of all axioms. In this phase we do the following:

1. count, for each symbol, the number of axioms in which it occurs;
2. store the set of pairs (s, A) such that s triggers A.

Note that this phase can be implemented in the time linear in the size of the set of axioms. It can be done by two traversals of the axioms (computing the trigger relation needs the number of occurrences).

In the second phase (which depends on the goal) we build the set of all triggered (or k-triggered, if k is given) axioms using the stored trigger relation. If the trigger relation is indexed on the first argument, the time complexity of the second phase is linear in the size of the resulting set of selected axioms and independent of the overall number of theory axioms. After the selection, the goal and the selected axioms are passed to a first-order theorem prover.

Separating the two steps of the algorithm provides an efficient way to treat collections of problems that share a large number of theory axioms. After preprocessing the shared theory axiomatisation, only the second phase is run on each of the problems, which allows us to avoid repeated execution of the first phase.

4 Variations

To turn the Sine selection on, one uses Vampire with the option

```
--sine_selection on
```

(the default value of this option is `off`). In the previous versions of Vampire we had two values: `axioms` and `included` instead of `on`. The value `axioms` considered as axioms the formulas marked as such in the TPTP language. The value `included` considered as axioms formulas coming from included files.[1] However, the value `included` is fragile since simple preprocessing of the input can change which formulas are included. In addition it turned out not to be good in practice, so in the current version we replaced these two values by a single one.

In the rest of this section we consider several variations of Sine selection. Each of these variations is implemented in Vampire as a parameter whose value can be set by the user. We present experimental evaluation of these parameters in Section 5.

4.1 Tolerance

Since our selection algorithm is incomplete, we introduced parameters changing the trigger function in various ways. One obvious problem with the selection is that it is very fragile with the respect to the number of axioms in which a symbol occurs, as already illustrated in Examples 3 and 4. Indeed, suppose a symbol s_1 occurs in (say) 7 axioms while s_2 occurs in 8 axioms. One can argue that s_1 and s_2 are, essentially, equally common. However, s_1 can trigger an axiom in which both s_1 and s_2 occur, while s_2 cannot trigger it.

To cope with this problem, we introduced a parameter called *tolerance*. The value of this parameter is a real number $t \geq 1$. It changes the trigger relation as follows.

Definition 5 (Trigger relation with tolerance). *Given the tolerance $t \geq 1$, define the relation trigger as follows: $trigger(s, A)$ iff for all symbols s' occurring in A we have $occ(s) \leq t \cdot occ(s')$.*

Compare this definition with Definition 2.

[1] The TPTP language classifies the input formulas into axioms, additional assumptions (hypotheses) and conjectures. Formulas can also be included from files using the TPTP directive include().

Example 6 (Sine selection with tolerance). Consider the set of axioms and the goal of Example 3. Assume the tolerance value is 1.5. This changes the trigger relation for two of the axioms as follows:

axiom	symbols
`subclass(X,Y)` $\wedge$ `subclass(Y,Z)` $\rightarrow$ `subclass(X,Z)`	`subclass`
`subclass(petrol,liquid)`	`petrol`
$\neg$`subclass(stone,liquid)`	`stone`
`subclass(beverage,liquid)`	`beverage,liquid`
`subclass(beer,beverage)`	`beer,beverage`
`subclass(guinness,beer)`	`guinness`
`subclass(pilsner,beer)`	`pilsner`

For the same goal `subclass(beer,liquid)`, the set of selected axioms becomes

```
subclass(X,Y) ∧ subclass(Y,Z) → subclass(X,Z)
subclass(beverage,liquid)
subclass(beer,beverage)
```

This set is sufficient to prove the goal. □

Note that the set of selected axioms is monotonic with regard to the value of tolerance: if we increase the value, all previously selected axioms will also be selected. For large enough values of tolerance, the set of selected axioms is simply the set of all relevant axioms in the sense of Section 2.1, because axioms become triggered by all the symbols that occur in them. For example, in Example 3 all axioms become selected when $t \geq 3$ and each symbol triggers all axioms in which it occurs when $t \geq 7$.

Having a fixed value of the tolerance parameter, we may perform axiom selection using the two-phase algorithm described in section 3.3. However, a likely scenario is that we will want to run several proof attempts with different values of the tolerance parameter. Using the basic two-phase algorithm, we would have to run the first phase of the algorithm for each value of the tolerance parameter. At the cost of a slight increase in the complexity, the algorithm may be modified, so that the first phase is run only once, allowing selection with arbitrary tolerance values. During the first phase, instead of storing a mapping from every symbol s to the set of axioms triggered by s, we store a mapping from s to the list of all pairs $(A_1, t_1), \ldots, (A_m, t_m)$ such that s triggers A_i if $t \geq t_i$. This list is ordered by the values of the t_i's. Ordering such lists can take $n \cdot log(n)$ time, so the complexity of the first phase slightly increases. However, the complexity of the second phase does not change: it is still linear in the size of selected axioms, since we only inspect the sublist of $(A_1, t_1), \ldots, (A_m, t_m)$ corresponding to the trigger relation.

To change the value of tolerance from the default value 1 to t, one uses Vampire with the option

```
--sine_tolerance t
```

4.2 Depth Limit

One can restrict the number of steps in computing the set of selected axioms so that it computes the set of all d-step triggered axioms. To this end, one can run Vampire with the option

```
--sine_depth d
```

The default value of `sine_depth` in Vampire is ∞. Evidently, the set of selected axioms is monotonic with regard to d: if we increase the value, all previously selected axioms will be selected, too.

4.3 Generality Threshold

The last modification of Vampire is based on the following idea: if a symbol s occurs in few axioms, then it triggers any axiom in which it occurs, even if the axiom contains symbols with fewer occurrences. To implement this, we fix some positive integer value $g \geq 1$ (called generality threshold) and modify the trigger relation as follows.

Definition 7 (Trigger relation with generality threshold). *Given the generality threshold $g \geq 1$, define the relation trigger as follows: $trigger(s, A)$ iff either $occ(s) \leq g$ or for all symbols s' occurring in A we have $occ(s) \leq occ(s')$.*

To turn the generality threshold in Vampire on, one can run Vampire with the option

```
--sine_generality_threshold g
```

The default value is 0 and the set of selected axioms is, evidently, monotonic with regard to g. If we set g to a large enough number, for example, the number of all axioms, then (similarly to setting a large value of the tolerance parameter) all the relevant axioms will be selected.

To use both a tolerance value t and a generality threshold value g, one should define the trigger relation as the union of the corresponding trigger relations. Namely, $trigger(s, A)$ iff either $occ(s) \leq g$ or for all symbols s' occurring in A we have $occ(s) \leq t \cdot occ(s')$.

5 Experiments

All experiments described in this section were carried out using a cluster of 64-bit quad core Dell servers having 12 GB RAM each.[2] Each of the runs used only one core and we never ran more than 3 tests in parallel on one computer to achieve the best performance.

The experiments were run on three benchmark suites taken from the TPTP library [11]. The library contains three different classes of very large problems:[3]

1. problems from the SUMO ontology [6]: `CSR075` to `CSR109`.
2. problems from the CYC knowledge base [4]; `CSR025` to `CSR074`.
3. problems from the Mizar library [9]: `ALG214` to `ALG234`, `CAT021` to `CAT037`, `GRP618` to `GRP653`, `LAT282` to `LAT380`, `SEU406` to `SEU451`, and `TOP023` to `TOP048`.

[2] The cluster was donated to our group by the Royal Society.

[3] These classes correspond to categories of the LTB division in the CASC competition [12].

Table 1. Average problem size information

problems	axioms	atoms	predicates	functions
SUMO	298,420	323,170	20	24,430
CYC	3,341,990	5,328,216	204,678	1,050,014
Mizar	44,925	332,143	2,328	6,115

Table 2. Selected formulas of CYC problems depending on the depth, tolerance and generality threshold

	$t = 1.0$		$t = 1.2$		$t = 1.5$		$t = 2.0$		$t = 3.0$		$t = 5.0$	
$d = 1$	29	1.17	35	1.09	41	1.05	47	1.02	60	1.02	72	1.01
$d = 2$	142	1.25	287	1.07	442	1.03	607	1.01	1027	1.00	1476	1.00
$d = 3$	505	1.32	937	1.13	1451	1.07	2484	1.02	5311	1.01	10482	1.01
$d = 4$	1784	1.41	3232	1.20	5716	1.10	11603	1.02	29963	1.01	69015	1.01
$d = 5$	4432	1.57	8870	1.27	16806	1.13	37599	1.03	110186	1.02	249192	1.04
$d = 7$	10698	2.16	25607	1.50	56337	1.21	150277	1.06	431875	1.09	832935	1.10
$d = \infty$	36356	28.37	495360	3.33	1310965	1.34	1562064	1.20	1822427	1.12	2057597	1.07

Each of these classes contains several different axiomatisations. To evaluate the size of the set of selected axioms and the number of iterations we only considered the largest axiomatisations in each class (that is, SUMO problems with the suffix +3, CYC problems with the suffix +6, and Mizar problems with the suffix +4. Table 1 contains information about sizes of these problems.

5.1 Generality Threshold

Table 2 shows how the number of selected formulas depends on the generality threshold. We considered the smallest possible value $g = 0$ and a sufficiently large value $g = 16$. In every column of the table we show on the left the number of axioms selected when $g = 0$ and on the right the number of axioms selected when $g = 16$ divided by the value on the left. The numbers are average over all CYC problems. One can see from the table that the largest increase (by the factor of 28.37) was achieved when the depth was unlimited and tolerance equal to 1. Predictably, when the tolerance grows, the percentage of additional axioms selected by the generality threshold value becomes smaller, since some axioms selected due to the large value of generality threshold become selected due to the high tolerance.

Our results have also shown that all the problems Vampire could solve, could also be solved with the value $g = 0$, so for the rest of this section we will focus on the remaining two options (depth and tolerance) and only consider the results obtained when the generality threshold was not used, that is, when $g = 0$. Therefore, the conclusion we can draw is that, although the generality threshold parameter is intended to cope with the problem of common relations, in practice it can be replaced by other parameters.

5.2 Selected Formulas

Table 2 shows the average numbers of selected axioms for the CYC problems, and table 3 shows these numbers for the Mizar and SUMO problems. Note that the numbers

Table 3. The number of selected formulas for the Mizar (left) and SUMO (right) problems

$d\backslash t$	1.0	1.2	1.5	2.0	3.0	5.0
1	4903	4911	4921	4936	4973	5038
2	5296	5395	5553	5823	6427	7743
3	6118	6451	7068	8280	10841	16337
4	6893	7556	9001	12176	18300	28878
5	7432	8517	11165	16945	26842	37284
7	7897	9991	15788	26203	36507	41443
∞	8047	15987	28353	35345	39389	41762

$d\backslash t$	1.0	1.2	1.5	2.0	3.0	5.0
1	12	13	14	16	21	28
2	70	82	115	158	272	654
3	188	230	372	762	1950	5980
4	316	470	942	3021	8720	23440
5	540	979	2417	8179	22644	52241
7	1027	2708	8517	24445	54958	97481
∞	1116	8361	26959	57322	82379	107926

Table 4. The minimal, average and maximal number of steps required to build the set of all triggered axioms as a function of tolerance

suite		$t = 1.0$	$t = 1.2$	$t = 1.5$	$t = 2.0$	$t = 3.0$	$t = 5.0$
	min	4	7	36	25	29	23
CYC	avg	19.3	102.7	44.3	29.9	33.0	25.1
	max	47	135	60	41	37	27
	min	6	7	19	15	14	10
Mizar	avg	9.5	27.5	25.4	19.7	15.0	12.3
	max	15	61	33	23	18	14
	min	3	5	4	6	13	11
SUMO	avg	7.3	15.3	20.8	19.8	16.6	12.6
	max	17	35	39	25	23	15

for the Mizar problems are essentially different from the SUMO and CYC problems. The number of selected Mizar axioms is large (over 4,000) even when the depth limit is set to 1, while it is relatively small for the CYC and SUMO problems. This is related to the fact that the Mizar problems usually have complex goals containing several assumptions and many symbols, while for the CYC and SUMO problems the goal is simple and usually is just an atomic formula with very few symbols. This is also one of the reasons why Mizar problems are much harder for all theorem provers using the Sine selection.

5.3 Number of Iterations

The next question we are interested in is the number of steps required by Sine selection to compute the set of all triggered relations. Table 4 contains statistics about the number of iterations. To our surprise, in some cases this number is very large (135 for CYC problems, 61 for Mizar problems, and 39 for SUMO problems). There is also no obvious pattern on how this parameter depends on the value of tolerance.

5.4 Essential Parameters

For experiments described in this subsection we considered all (not only largest) SUMO, CYC and Mizar problems.

Since our main aim is to automatically prove (hard) theorems, the most important questions related to the use of Sine selection are the following:

Table 5. Problems solved with and without Sine selection

atoms	only with Sine	only without Sine	together
10,000	243	64	721
20,000	217	10	542
40,000	208	7	464
80,000	187	3	373
160,000	138	1	243
320,000	80	1	168
640,000	50	0	100
1,280,000	50	0	50
rating 1	232	25	402

1. How powerful is the selection method?
2. Which of the parameters (and ranges of values for these parameters) are essential in practice?

The first question cannot be answered in a simple way. On the one hand, we have strong evidence that the method is very powerful. To support this, consider Table 5. It shows the number of all the TPTP 4.0.1 CYC, Mizar, and SUMO problems solved with some Sine selection and without it, depending on the size of the problem measured as the number of atoms in it. The last row shows these numbers for the problems having TPTP rating 1. A problem has TPTP rating 1 if it was previously unsolved by all provers, including the previous versions of Vampire. For example, among the problems having 80,000 or more atoms, 373 problems were solved by Vampire all together. 138 problems could only be solved with the help of Sine selection, while only 3 problems out of 373 could not be solved with Sine selection.

When a problem is not solved, we do not know why Vampire (or any other prover) fails to prove it. This can be for at least the following three reasons, of which two are directly related to the power of our selection method:

1. the set of selected axioms can be insufficient to prove the goal;
2. the set of selected axioms is too large, which prevents theorem provers from success;
3. the problem is very hard even for small sets of axioms sufficient to prove the goal.

Let us now investigate which parameters and their values are essential for Vampire. As we pointed out, it turned out that the generality threshold parameter can be dropped without any effect on the set of problems solved by Vampire. It turned out that both the tolerance and depth limit are very essential. To show this, we used our database of proofs found by Vampire, which was generated using about 70 CPU years of run time and now contains about 575,000 results, of which over 43,000 are related to the mentioned benchmark suite.

We selected problems having less than 10 solutions in the database. The reason to use the number of solutions as a criterion was that problems with few solutions are believed to be harder. Also, such problems can be solved only with a small subset of possible values for the various Vampire parameters.

Table 6. The sine depth range for solved hard problems

range	CYC	Mizar	SUMO
$1-1$		16	
$1-2$		10	
$1-3$		5	
$1-4$		3	
$1-5$		2	
$1-10$		1	
$1-\infty$	15	107	6
$2-2$		21	
$2-3$		12	
$2-4$		3	
$2-5$		6	
$2-7$		1	
$2-10$		1	
$2-\infty$	21	39	4
$3-3$		1	
$3-4$		1	
$3-\infty$	6	1	1
$4-4$		1	
$4-\infty$	6		
$5-\infty$	3		
$10-\infty$	3		1
total	51	231	12

Table 7. The sine tolerance range for solved hard problems

range	CYC	Mizar	SUMO
$1.0-1.0$		3	
$1.0-1.2$		4	
$1.0-1.5$		12	1
$1.0-2.0$		17	
$1.0-3.0$		19	
$1.0-5.0$	49	155	11
$1.2-1.5$		2	
$1.2-2.0$		1	
$1.2-3.0$		1	
$1.2-5.0$	2	1	
$1.5-5.0$		1	
$2.0-2.0$		1	
$2.0-3.0$		7	
$2.0-5.0$		1	
$3.0-5.0$		3	
$5.0-5.0$		2	
total	51	231	12

This selection resulted in 51 CYC problems, 231 Mizar problems and 12 SUMO problems. For each of these problems we checked which parameter values solve these problems. More precisely, we took their known solutions, changed the depth and tolerance parameters and checked which of the changes still solve the problem. The results are summarised in Tables 6 and 7. The table cells show the number of problems which can be solved by the given range of values. We do a projection on the possible values of the parameter that is not present in the table (tolerance in the case of Table 6 and depth limit in Table 7). For example, the third row in Table 7 means that there were 12 selected Mizar problems and 1 selected SUMO problem that could be solved with some values of the depth limit and only with the values of tolerance between 1.0 and 1.5.

Let us first analyse the depth limit in Table 6. For the evaluation we used the following values of depth limit: 1, 2, 3, 4, 5, 7, 10 and ∞. The first observation is that this parameter is, indeed, very important. For example, there were 39 Mizar problems that could be solved with only one value of this parameter (1, 2, 3 or 4). For both CYC and SUMO collections setting the depth to ∞ is always a good strategy. On the contrary, only 147 out of 231 solved Mizar problems (and only 30 of 64 the largest Mizar problems) could be solved with this setting.

Next, let us analyse the tolerance in Table 7. For the evaluation we used the following values of tolerance: 1.0, 1.2, 1.5, 2.0, 3.0, 4.0, and 5.0. It turned out that this parameter is also very important. However, the behaviour of solutions depending on this parameter is more stable than for the depth parameter: only 6 Mizar problems could be solved with exactly one value of the tolerance and 155 Mizar problems out of 231 were solved with all the values we tried. Among the largest hard Mizar problems, only 36 out of 64 were solved with all the tried values of tolerance. 11 out of 12 SUMO problems and 49 out of 51 CYC problems could be solved with any value of the tolerance and all CYC problems could be solved with the value 1.2 or higher.

6 Competition Performance

Our axiom selection algorithm was used by several systems participating in the Large Theory (LTB) division of recent CASC competitions.

The Sine selection was introduced in 2008, at the CASC-J4 [10] competition. The only participant that used our algorithm was the SInE theorem prover. It has won the division by solving 88 out of 150 problems, which was 12 problems ahead of the second best participant.

In 2009, at the CASC-22 [12] competition, four out of seven participants were using our selection algorithm, and these four participants ended up at the first four positions, solving 69 to 35 problems out of 100, while the best participant not using our algorithm solved only 18 problems.

In 2010, at the CASC-J5 competition, five out of seven systems were using our algorithm as the only axiom selection algorithm, including the winner (Vampire). The second best ranked system (Currahee) used our selection algorithm as one of possible selection algorithms.

7 Related Work

In our algorithm we maintain the set of selected axioms (starting from goal), and select new axioms that are relevant to the goal step by step. The lightweight relevance filtering algorithm [5] shares this approach, but instead of a trigger relation, which is used by our algorithm, it selects an axiom if certain percentage of its symbols appears in the already selected axioms. This method also penalizes common symbols—the more often a symbol appears in the problem, the less impact its appearance in an axiom has.

Several other algorithms use some measure of distance in a graph of axioms, in order to determine which axioms are relevant to the goal. The contextual relevance filtering [1] and the syntactic relevance measure [13] compute the weighted graph between axioms with weights based on the number of shared symbols (taking into account their commonness). The latter does not use the distance from conjecture to select relevant axioms, but to order them for further (semantic) processing. The relevance restriction strategy described in [7] connects two clauses in the graph if they have unifiable literals. The paper also examines the suitability of different graph distance measures.

Semantic algorithms are another group of axiom selection algorithms that use a model of currently selected axioms to guide the selection of new ones. To this group belong the Semantic Relevance Axiom Selection System [13] and the algorithm for semantic selection of premises [8].

Yet another approach is taken in the latent semantic analysis [1], which uses a technique for analysing relationships between documents. Each formula is considered to be a document, and formulas with strong relationship toward the goal are selected. The MaLARea system [16] uses machine learning on previous proofs in the theory to estimate which theory axioms are likely to contribute to proofs of new problems.

In [15] the benefit of our axiom selection algorithm for reasoning on the Mizar Mathematical Library [14] is examined.

8 Conclusion

We defined the Sine selection used in the theorem prover Vampire and several other theorem provers to select axioms potentially relevant to the goal. We formalised the Sine selection as a family of *trigger-based selection algorithms*. We showed that all the existing axiom selection parameters in Vampire can be formalised as a special case of such algorithms.

We also discussed, using extensive experiments over all TPTP problems with large axiomatisations, the effect of various parameter values on the size of the selected set of axioms, the number of iterations of the algorithm, and solutions of hard TPTP problems.

We also added a new mode to Vampire to make others able to experiment with our axiom selection. If Vampire is run in a new *axiom selection mode*, it does not try to prove the problem but only selects axioms according to the user-given options and outputs the selected axioms and the goal in the TPTP format. This mode can be invoked by using `vampire --mode axiom_selection`.

Acknowledgements

The authors would like to thank Josef Urban for helpful suggestions, and for his support and supervision of the first author during the first year of the work on the Sine algorithm.

References

1. Roederer, A., Puzis, Y., Sutcliffe, G.: divvy: An ATP meta-system based on axiom relevance ordering. In: Schmidt, R.A. (ed.) CADE-22. LNCS, vol. 5663, pp. 157–162. Springer, Heidelberg (2009)
2. Armando, A., Baumgartner, P., Dowek, G. (eds.): IJCAR 2008. LNCS (LNAI), vol. 5195. Springer, Heidelberg (2008)
3. Hoder, K., Kovács, L., Voronkov, A.: Interpolation and symbol elimination in vampire. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS, vol. 6173, pp. 188–195. Springer, Heidelberg (2010)
4. Lenat, D.B.: CYC: A large-scale investment in knowledge infrastructure. Communications of the ACM 38(11), 33–38 (1995)
5. Meng, J., Paulson, L.C.: Lightweight relevance filtering for machine-generated resolution problems. J. Applied Logic 7(1), 41–57 (2009)
6. Niles, I., Pease, A.: Towards a standard upper ontology. In: FOIS, pp. 2–9 (2001)
7. Plaisted, D.A., Yahya, A.H.: A relevance restriction strategy for automated deduction. Artif. Intell. 144(1-2), 59–93 (2003)
8. Pudlak, P.: Semantic selection of premisses for automated theorem proving. In: Sutcliffe, G., Urban, J., Schulz, S. (eds.) ESARLT. CEUR Workshop Proceedings, vol. 257, pp. 27–44. (2007) CEUR-WS.org
9. Rudnicki, P. (ed.): An overview of the mizar project, pp. 311–332. University of Technology, Bastad (1992)

10. Sutcliffe, G.: CASC-J4 the 4th IJCAR ATP system competition. In: Armando (ed.) [2], pp. 457–458
11. Sutcliffe, G.: The tptp problem library and associated infrastructure. J. Autom. Reasoning 43(4), 337–362 (2009)
12. Sutcliffe, G.: The cade-22 automated theorem proving system competition - casc-22. AI Commun. 23(1), 47–59 (2010)
13. Sutcliffe, G., Puzis, Y.: Srass - a semantic relevance axiom selection system. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 295–310. Springer, Heidelberg (2007)
14. Urban, J.: Mptp 0.2: Design, implementation, and initial experiments. J. Autom. Reasoning 37(1-2), 21–43 (2006)
15. Urban, J., Hoder, K., Voronkov, A.: Evaluation of automated theorem proving on the mizar mathematical library. In: Fukuda, K., van der Hoeven, J., Joswig, M., Takayama, N. (eds.) ICMS 2010. LNCS, vol. 6327, pp. 155–166. Springer, Heidelberg (2010)
16. Urban, J., Sutcliffe, G., Pudlák, P., Vyskocil, J.: Malarea sg1- machine learner for automated reasoning with semantic guidance. In: Armando (ed.) [2], pp. 441–456.